

オープンソースのセキュリティ懸念について（ある顛末）

基調講演 @ Osaka NDS Embedded Cross Online Forum #13

宗像尚郎

Automotive Grade Linux Advisory board member
Linux Foundation board member

2021-7-9

自己紹介

コミュニティ活動 ← 今日はこちらの帽子

- Linux Foundation Board Director
- AGL Advisory Board
- Genivi Board
- yocto project advisory board
- Xen FUSA SIG メンバー
- Chromium Upstream 開発支援

企業活動 (ルネサスエレクトロニクス)

- R-Car Linux upstream kernel 開発
- Vehicle-to-Cloud Solition 開発
- SW First 活動への支援
- Sustainable SW サポート提供
- 社内開発者育成支援
- 顧客開発プロジェクト支援

企業の **SW** 開発者と **OSS** 開発コミュニティの橋渡しを多面的にサポートしてきました

サイバーセキュリティと OSS の関わり

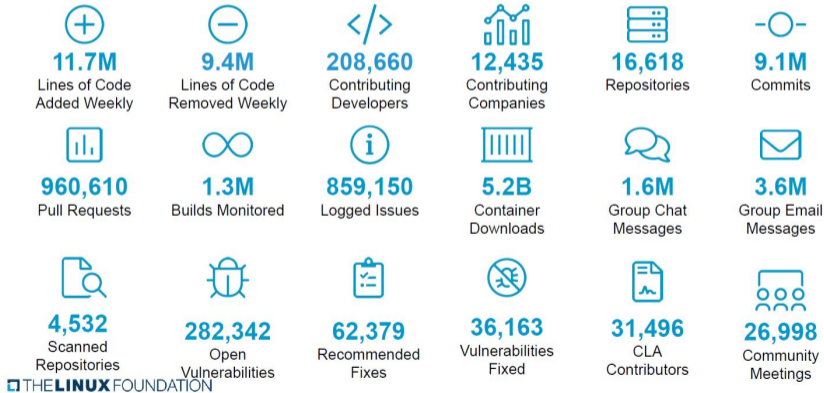
オープンソースは既に **重要な社会インフラ基盤** となっている

In the first two months of 2021 we are adding 2-3 new projects per week: amongst the most important shared technology on earth (and mars). 

Security	       
Networking	      
Cloud	      
Automotive	  
Blockchain	     
Edge/IoT	       
Web	       
AI	       
Film	      
CI/CD	        
Energy	       
Hardware	    
Standards	      

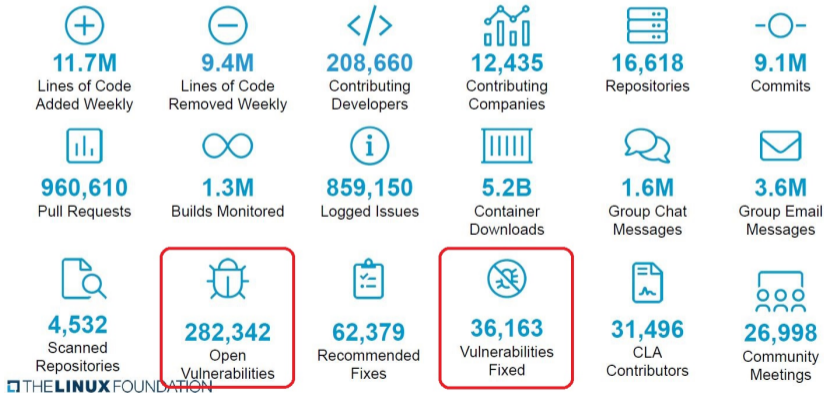
オープンソースは既に **重要な社会インフラ基盤** となっている

We continue to scale our communities on every level by automating processes, creating innovative developer tools, and focusing on community value.



オープンソースは既に **重要な社会インフラ基盤** となっている

We continue to scale our communities on every level by automating processes, creating innovative developer tools, and focusing on community value.



サイバーセキュリティ対策強化 が最重要な社会課題となっている

THE WHITE HOUSE



Administration Priorities COVID-19 Briefing Room Español MENU

BRIEFING ROOM

Executive Order on Improving the Nation's Cybersecurity

MAY 12, 2021 • PRESIDENTIAL ACTIONS



By the authority vested in me as President by the Constitution and the laws of the United States of America, it is hereby ordered as follows:

Section 1. Policy. The United States faces persistent and increasingly sophisticated malicious cyber campaigns that threaten the public sector, the private sector, and ultimately the American people's security and privacy.

The Federal Government must improve its efforts to identify, deter, protect against, detect, and respond to these actions and actors. The Federal Government must also carefully examine what occurred during any major cyber incident and apply lessons learned. But cybersecurity requires more than government action. Protecting our Nation from malicious cyber actors requires the Federal Government to partner with the private sector. The private sector must adapt to the continuously changing threat environment, ensure its products are built and operate securely, and partner with the

<https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order>

Linux Foundation のサイバーセキュリティに対する取り組み

How LF communities enable security measures required by the US Executive Order on Cybersecurity



Our communities take security seriously and have been instrumental in creating the tools and standards that every organization needs to comply with the recent US Executive Order

<https://www.linuxfoundation.org/blog/how-lf-communities-enable-security-measures-required->

コミュニティ起点の対策強化の取り組み

- Open Source Security Foundation (OpenSSF)
 - 脆弱性情報開示
 - Best Practice WG
 - 検出ツールの開発、改善
 - デジタルアイデンティティ認証
- SPDX
 - SBOM (SW Bill of Material)
 - SW Supply Chain 管理スキーム

オープンソースとサイバーセキュリティリスク (二つの見解)

OSS はセキュリティリスクが心配

- 「安かろう悪かろう」の論理
- 開発者の素性が分からない
- コードの品質レベルがまちまち
- アタックサーフェスが丸見え
- サポート／メンテナンススキーム不在
- 問題発生時の責任当事者が不明確

OSS の方が信頼できる

- トレーサビリティ
 - ソースコードへのアクセス
 - 変更履歴が開示されている
- "Given enough eyeballs, all bugs are shallow" (Linus's Law)
- サステナビリティ
 - コードがあればメンテは継続できる

さすがに以前のような乱暴な主張 (**OSS= 悪** が前提) は減ってきているが

実態としては 利用実績の多いものほど脆弱性報告数が多くなる

CVE Details

The ultimate security vulnerability datasource

(e.g.: CVE-2009-1234 or 2010-1234 or 20101234)

 Search
View CVE

Log In Register

Vulnerability Feeds & Widgets ^{NEW} www.itsecdb.com

- [Switch to https://](#)
- [Home](#)
- Browse :**
- [Vendors](#)
- [Products](#)
- [Vulnerabilities By Date](#)
- [Vulnerabilities By Type](#)
- Reports :**
- [CVSS Score Report](#)
- [CVSS Score Distribution](#)
- Search :**
- [Vendor Search](#)
- [Product Search](#)
- [Version Search](#)
- [Vulnerability Search](#)
- [By Microsoft References](#)
- Top 50 :**
- [Vendors](#)
- [Vendor Cvss Scores](#)
- [Products](#)
- [Product Cvss Scores](#)
- [Versions](#)
- Other :**

CVSS Score Distribution For Top 50 Products By Total Number Of "Distinct" Vulnerabilities

	Product Name	Vendor Name	Number of Total Vulnerabilities	# Of Vulnerabilities										Weighted Average	% Of Total									
				0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9+		0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9+
1	Debian Linux	Debian	5078	86	255	126	1427	1055	969	932	14	234	6.20	0	1	5	2	28	21	19	18	0	5	
2	Android	Google	3651	33	363	40	913	261	294	734	33	680	6.90	0	1	10	1	25	10	8	20	1	24	
3	Ubuntu Linux	Canonical	2984	43	189	92	889	522	469	545	7	280	6.20	0	1	6	3	30	17	16	18	0	8	
4	Mac Os X	Apple	2759	1	23	194	24	492	336	654	502	11	522	6.90	0	1	7	1	18	12	24	18	0	19
5	Linux Kernel	Linux	2668	1	104	427	67	856	149	211	719	9	188	5.80	0	4	16	3	32	6	8	27	0	5
6	Iphone Os	Apple	2300	26	159	19	438	226	817	217	6	380	6.80	0	1	7	1	19	10	36	9	0	17	
7	Windows 10	Microsoft	2260	86	317	32	641	75	242	533	7	321	6.30	0	4	14	1	28	3	11	24	0	14	
8	Chrome	Google	2161			16	622	221	697	418	1	166	6.70	0	0	1	0	29	10	32	19	0	9	
9	Windows Server 2016	Microsoft	2021	82	290	32	594	78	216	451	7	221	6.20	0	4	14	2	29	4	11	22	0	13	
10	Windows Server 2008	Microsoft	1973	79	320	15	396	69	199	529	12	454	6.90	0	4	11	1	20	3	10	27	1	23	
11	Fedora	Fedoraproject	1959	23	102	46	550	465	388	305	6	28	6.10	0	1	5	2	28	24	20	16	0	4	
12	Firefox	Mozilla	1916	8	52	4	408	340	259	335	1	509	7.20	0	0	3	0	21	18	14	17	0	27	
13	Windows 7	Microsoft	1854	77	225	12	397	53	187	492	10	491	6.80	0	4	12	1	21	3	10	27	1	22	
14	Windows Server 2012	Microsoft	1728	85	240	19	359	77	185	416	8	430	6.50	0	5	14	1	21	4	11	24	0	20	

<https://www.cvedetails.com/top-50-product-cvssscore-distribution.php>

サイバーセキュリティ対策はアカデミアでも重要な研究テーマに

Check It Again: Detecting Lacking-Recheck Bugs in OS Kernels

Wenwen Wang, Kangjie Lu, and Pen-Chung Yew
University of Minnesota, Twin Cities

ABSTRACT

Operating system kernels carry a large number of security checks to validate security-sensitive variables and operations. For example, a security check should be embedded in a code to ensure that a user-supplied pointer does not point to the kernel space. Using security-checked variables is typically safe. However, in reality, security-checked variables are often subject to modification after the check. If a recheck is lacking after a modification, security issues may arise. e.g., adversaries can control the checked variable to launch critical attacks such as out-of-bound memory access or privilege escalation. We call such cases lacking-recheck (LRC) bugs, a subclass of TOCTTOE bugs, which have not been explored yet.

In this paper, we present the first in-depth study of LRC bugs and develop LRSan, a static analysis system that systematically detects LRC bugs in OS kernels. Using an inter-procedural analysis and multiple new techniques, LRSan first automatically identifies security checks, critical variables, and uses of the checked variables, and then reasons about whether a modification is present after a security check. A case in which a modification is present but a recheck is lacking is an LRC bug. We apply LRSan to the latest Linux kernel and evaluate the effectiveness of LRSan. LRSan reports thousands of potential LRC cases, and we have confirmed 19 new LRC bugs. We also discuss patching strategies of LRC bugs based on our study and bug fixing experience.

CCS CONCEPTS

Security and privacy → Operating systems security.

KEYWORDS

OS Kernel Bug Mining; Check; Lacking-Recheck; Error Code; TOCTTOE; Static Analysis

ACM Reference Format:

Wenwen Wang, Kangjie Lu, and Pen-Chung Yew. 2018. Check It Again: Detecting Lacking-Recheck Bugs in OS Kernels. In *2018 ACM SIGPLAN Conference on Computer and Communications Security (CCS '18)*, October 20–19, 2018, Toronto, ON, Canada. ACM, New York, NY, USA, 19 pages. <https://doi.org/10.1145/3277413.3277444>.

1 INTRODUCTION

Operating system (OS) kernels, as the core of computer systems, play a critical role in managing hardware and system resources. They also provide services in the form of system calls to user space.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made for commercial profit or for advertising or promotional purposes, not for resale, and that the copyright notice, this notice, and the full citation are included on the first page. Copyright for this work is owned by ACM, Inc. All rights reserved. This work is licensed under a Creative Commons Attribution 4.0 International License. For more information, see <http://creativecommons.org/licenses/by/4.0/>.

CCS '18, October 20–19, 2018, Toronto, ON, Canada.
© 2018 Copyright held by the owner/authors. Publication rights licensed to ACM.
0739-4061/18/10-19-19...\$15.00
<https://doi.org/10.1145/3277413.3277444>

code. In order to safely manage resources and to stop attacks from the user space, OS kernels carry a large number of security checks that validate key variables and operations. For instance, when the Linux kernel fetches a data pointer, `ptr`, from the user space for a memory write, it uses `access_ok(VERIFY_WRITE, ptr, size)` to check if both `ptr` and `ptr+size` point to the user space. If the check fails, OS kernels typically return an error code and stop executing the current function. Such a check is critical because it ensures that a memory write from the user space will not overwrite kernel data. Common security checks in OS kernels include permission checks, bound checks, return value checks, NULL pointer checks, etc.

A security-checked variable should not be modified before being used. Otherwise, the security check is rendered ineffective. However, in practice, due to the existence of unusual execution flows and implicit modifications, a checked variable may be further modified unintentionally because of developers' ignorance of such an occurrence. If a recheck is not enforced after the modification, potential violations against the security check may occur, leading to critical security issues such as out-of-bound memory access and privilege escalation. We define such a case as an LRC bug: a variable is modified after being security checked, but it lacks a recheck after the modification. In other words, an LRC bug exists when two conditions are satisfied: (1) the execution has a sequence of three operations—security checking a variable, modifying the variable, and using the variable; (2) a recheck does not exist between the modification and a use of the variable.

LRC bugs can be considered a subclass of time-of-check-to-time-of-use (TOCTTOE) bugs because both of them refer to the abstract concept of modification after check but before use. However, LRC bugs differ from traditional bugs such as missing-check bugs [35, 47], double-fetch bugs [16, 33, 41, 46], and atomicity-violation bugs [14, 15, 27, 42]. Intuitively, LRC bugs are different from missing-check bugs [35, 47]. A check is completely absent in a missing-check bug, which by definition will not be identified as an LRC bug. First, for LRC bugs, checks are not “missing” at all. Existing missing-check detectors [35, 47] does not warn against modifications to checked variables—so long as at least a check has been performed. We also differentiate LRC bugs from double-fetch bugs. Double-fetching is a common programming practice for the performance reason. It is not a bug by itself if recheck is enforced after the second copy. By contrast, LRC bugs are actual check-bypassing bugs that violate security checks. Moreover, LRC bugs target general critical data, not just the ones from the user space. To compare with atomicity-violation bugs that exist in only concurrent programs, an LRC bug can exist in a single-threaded program—a thread itself may modify a security-checked variable. We will discuss in detail about the characteristics of LRC bugs by comparing them to traditional bugs in §6.

We find that LRC bugs are common in OS kernels for several reasons. First, execution paths from a security check of a variable to a use of this variable can be fairly long and complicated, especially

Linux kernel bug-fix research @UMN

- UMN=University of Minnesota
- 2018年からOSセキュリティ対策研究をスタート
- Linux kernel ソースコードがターゲット
- Kangjie Lu 助教授が博士課程の学生を指導
- 約 400 件の bug-fix パッチ を投稿した実績あり
- パッチ投稿の経緯と成果を 学術論文 として発表
- IEEE などの学会で多数の発表実績あり

<https://www-users.cs.umn.edu/~kjlw/papers/lrsan.pdf>

ミネソタ大 (MNU) のコミュニティ出禁事件が大々的に報じられる事態に

Linuxカーネルに意図的にバグを混入したとして大学にコミュニティ出禁措置



オープンソースソフトウェアの脆弱(ぜいじゃく)性に関する論文の執筆のため、Linuxカーネルに既知のバグを含むパッチを送信したことを理由に、ミネソタ大学に対して「Linuxカーネル開発への貢献の禁止」、つまり出禁措置が行われました。

Linux bans University of Minnesota for sending buggy patches in the name of research [Update] - Neowin
<https://www.neowin.net/news/linux-bans-university-of-minnesota-for-sending-buggy-patches-in-the-name-of-research/>

<https://gigazine.net/news/20210422-linux-ban-university/>

Kangjie Lu @ UMN

Kangjie Lu Research Publication Teaching Advising Experience Service



Kangjie Lu

Assistant Professor

Department of Computer Science & Engineering
University of Minnesota

Office: 5-217 Keller Hall, 200 Union St SE Minneapolis, MN 55455

Email: kju@umn.edu

[Google scholar](#) | [GitHub](#)



I am an assistant professor in the Computer Science & Engineering Department of the University of Minnesota--Twin Cities. I research and teach systems security. My primary research lies at the intersection of security, operating systems, program analysis, and compilers. My research also occasionally involves machine learning and computer architecture. I earned my Ph.D. in Computer Science from Georgia Tech in 2017.

I'm looking for Ph.D. students, a postdoc, and visiting students. If you are interested in systems and security, please feel free to contact me! See [details](#).

The "hypocrite commits" work

On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits

- 11/21/2020: [Paper accepted](#)
- 12/15/2020: [Clarifications](#)
- 04/21/2021: The Linux incident happened due to superfluous patches from Aditya Pakki for a new bug-finding project
- 04/24/2021: [Open letter to Linux](#)
- 04/26/2021: [Paper withdrawn](#)
- 04/27/2021: Full disclosure of case study
- 04/27/2021: [Department response to the Linux Foundation](#): **The three hypocrite patches never were intended to be added to code, and, in fact, were not added.**
- 05/05/2021: [The Linux Technical Advisory Board report](#): **No malicious or bad-faith code found in the re-reviewed UMN patches.**

<https://www-users.cs.umn.edu/~kju/>

ある博士課程学生の疑問「はたして OSS 開発者は信頼できるのか？」

デベロッパーに起因するリスク

- OSS だと誰でもコードを投稿できるので、開発者のスキル等が不安
- 間違ってバグ（セキュリティリスク）を入れてしまう可能性がある
- さらに **故意にセキュリティリスクをこっそり入れることもできるのでは？**

メンテナーに起因するリスク

- **潜在的なセキュリティホール対策のパッチを受け付けない？**（Greg 曰く “It must fix a real bug that bothers people (not a, “This could be a problem...” type thing” in `stable_kernel_rules`)
- **静的解析ツール** などが十分活用されていなくて人間の目に頼っている？

バグ対策に見せかけ故意に脆弱性を入れる” **Hypocrite Commit**” 実験を思いつく

Qiushi Wu

Qiushi Wu

Ph.D. student in the Department of Computer Science & Engineering, at the University of Minnesota (Twin Cities)

Email: wu000273 at umn.edu

Office: 5-248 Keller Hall, 200 Union St SE Minneapolis, MN 55455

I am a Ph.D. student in the Computer Science & Engineering Department at the University of Minnesota, advised by professor Kangjie Lu. I received my undergraduate B.A. in the Information Science & Engineering Department of the University of Science and Technology of China in 2018. My primary research interest is applications of program analysis techniques on operating systems such as the Linux kernel. [Google scholar](#) | [CV](#)

Research

My research aims to protect widely used systems programs such as operating systems (OS) kernels with billions of users from security and reliability issues. Specifically, my research includes: (1) developing fundamental techniques that enable precise and scalable program analysis and (2) studying and detecting critical security bugs in foundational programs. In the past two years, I have designed and implemented several automated analysis tools, which scale precise symbolic execution to OS kernels with 27 million lines of code and detected hundreds of security bugs in multiple widely used systems such as the Linux kernel and the OpenSSL library; these works have been published at prestigious conferences including IEEE S&P, USENIX Security, NDSS, ESORICS, and etc. My works are impactful. On the one hand, the precise and scalable symbolic execution serves as a foundational technology that could benefit a variety of areas such as software engineering, systems, and compilers. On the other hand, my works are able to find a large number of critical security bugs in widely used programs with billions of users, thus can improve the security of computer systems, protect the integrity of user data and the privacy of users.

Publications

2021

- On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits [PDF] [FAQ] [Withdrawal-Letter] [Case-study disclosure] [Open letter to Linux]
Qiushi Wu, and Kangjie Lu.

<https://qiushiwu.github.io/>

2020 年 8 月～2021 年 5 月

2020-08-04: "Hypocrite Commits 1/5"

LKML Archive on lore.kernel.org

search [help](#) / [color](#) / [Atom feed](#)

From: James Bond <jameslouisebond@gmail.com>
To: jameslouisebond@gmail.com
Cc: "Gustavo A. R. Silva" <gustavo@embeddedor.com>,
Greg Kroah-Hartman <gregkh@suse.de>,
Mike Waychison <mikew@google.com>,
linux-kernel@vger.kernel.org
Subject: [\[PATCH\] firmware: dmi-sysfs: Add clean-up operations to fix refcount leak](#)
Date: Tue, 4 Aug 2020 13:36:49 -0500
Message-ID: <20200804183650.4024-1-jameslouisebond@gmail.com> ([raw](#))

According to the documentation of function `kobject_init_and_add()`, when this function returns an error, `kobject_put()` must be called to properly clean up the memory associated with the object.

Fixes: [925a1da7477f](#) ("firmware: Break out system_event_log in dmi-sysfs")
Signed-off-by: James Bond <jameslouisebond@gmail.com>

[drivers/firmware/dmi-sysfs.c](#) | 4 +++-
1 file changed, 3 insertions(+), 1 deletion(-)

<https://lore.kernel.org/lkml/20200804183650.4024-1-jameslouisebond@gmail.com/>

2020-08-20: "Hypocrite Commits 3/5"

LKML Archive on lore.kernel.org

search [help](#) / [color](#) / [Atom feed](#)

From: [George Acosta <acostag.ubuntu@gmail.com>](mailto:acostag.ubuntu@gmail.com)

To: acostag.ubuntu@gmail.com

Cc: [Herbert Xu <herbert@gondor.apana.org.au>](mailto:herbert@gondor.apana.org.au),
"David S. Miller" <davem@davemloft.net>,
[Christophe JAILLET <christophe.jaillet@wanadoo.fr>](mailto:christophe.jaillet@wanadoo.fr),
"Gustavo A. R. Silva" <gustavo@embeddedor.com>,
[Phani Kiran Hemadri <phemadri@marvell.com>](mailto:phemadri@marvell.com),
linux-crypto@vger.kernel.org, linux-kernel@vger.kernel.org

Subject: [\[PATCH\] crypto: cavium/nitrox: add an error message to explain the failure of pci request mem regions](#)

Date: Thu, 20 Aug 2020 22:12:08 -0500

Message-ID: <20200821031209.21279-1-acostag.ubuntu@gmail.com> ([raw](#))

Provide an error message for users when `pci_request_mem_regions` failed.

Signed-off-by: [George Acosta <acostag.ubuntu@gmail.com>](mailto:acostag.ubuntu@gmail.com)

[drivers/crypto/cavium/nitrox/nitrox_main.c](#) | 1 +
1 file changed, 1 insertion(+)



<https://lore.kernel.org/lkml/20200821031209.21279-1-acostag.ubuntu@gmail.com/>

2020-08-27: Linux kernel 開発コミュニティへの公開質問

Date Thu, 27 Aug 2020 20:27:30 +0200
From Greg Kroah-Hartman <>
Subject Re: Some questions about the patching process



On Thu, Aug 27, 2020 at 12:34:57PM -0500, Qiushi Wu wrote:

> Dear Linux maintainers:

>

> I'm Qiushi Wu, a Ph.D. student from the secure and reliable systems
> research group at the University of Minnesota. Our group strives to improve
> the security and reliability of the Linux kernel and has contributed quite
> a number of patches. We appreciate the openness of the Linux community, but
> would also like to discuss some questions about the patching process. It
> would be great if you could let us know your thoughts.

>

> We recently found that minor patches such as one fixing a memory leak may
> introduce more critical security bugs like double free. Sometimes the
> maintainers can capture the introduced security bugs, sometimes not. This
> is understandable because the introduced bugs can be subtle and hard to
> catch due to the complexity. We are more concerned when "bad" submitters
> intentionally and stealthily introduce such security bugs via seemingly
> good patches, as they indeed have a chance to get the actually bad patches
> accepted. This is not impossible because people have incentives---to plant
> a vulnerability in a targeted driver, to get bounty rewards, etc.

<https://lkml.org/lkml/2020/8/27/1197>

2020-08-27: Linux kernel 開発コミュニティへの公開質問

```
> Based on our patching experience, we observe several things related to the  
> risks.  
>  
> 1. Linux allows anyone to submit a patch because it is an open community.  
>  
> 2. Maintainers tend to not accept preventive patches, i.e., a patch for a  
> bug that is not really there yet, but it can be likely formed in the  
> future.
```

How can you create a patch to prevent a bug that is not present?

But no, this is not true, look at all of the kernel hardening features added to the kernel over the past 5+ years. Those are specifically to help handle the problem when there are bugs in the kernel, so that "bad things" do not happen when they occur.

```
> 3. The patch review is mainly manual, so sometimes the introduced security  
> bugs could be missed.
```

We are human, no development process can prevent this.

```
> We would like to know how the Linux maintainers think about these risks.
```

I think 2. is wrong, so it's not a risk.

And how is 1. a "risk"?

And of course, 3., humans, well, what can you do about them? :)

2020-08-27: Linux kernel 開発コミュニティへの公開質問

> We
> would like to know if maintainers have some methods and tools (such as
> Smatch, Syzbot?) to mitigate these potential issues. We are happy to
> discuss these issues and hope our observations could raise some awareness
> of them.

How do you "raise awareness" among a developer community that is 4000
people each year (1000 are new each year), consisting of 450+ different
companies?

And yes, we have lots of tools, and run them all the time on all of our
public trees constantly. And they fix things before they get merged and
sent out to the rest of the world.

So what specific things are you wanting to discuss here?

thanks,

greg k-h

2020-11-21: IEEE Symposium on Security and Privacy が論文採択

On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits

Qiushi Wu and Kangjie Lu
University of Missouri
{wu00273, lu}@missouri.edu

Abstract—Open source software (OSS) has thrived since the founding of Open Source Initiative in 1998. A prominent example is the Linux kernel, which has been used by numerous major software vendors and empowers billions of devices. The higher availability and lower costs of OSS boost its adoption, while its openness and flexibility enable quicker innovation. More importantly, the OSS development approach is believed to produce more reliable and higher-quality software since it typically has thousands of independent programmers testing and fixing bugs of the software collaboratively.

In this paper, we instead investigate the insecurity of OSS from a critical perspective—the feasibility of stealthily introducing vulnerabilities in OSS via hypocrite commits (i.e., seemingly beneficial commits that in fact introduce other critical issues). The introduced vulnerabilities are critical because they may be stealthily exploited to impact massive devices. We first identify three fundamental reasons that allow hypocrite commits: (1) OSS is open to anyone, no anyone from anywhere, including malicious ones, can submit patches; (2) Due to the overwhelming patches and performance issues, it is impractical for maintainers to accept preventive patches for “unknown vulnerabilities”; (3) OSS like the Linux kernel is extremely complex, so the patch-review process often misses introduced vulnerabilities that involve complicated semantics and require. We then systematically study hypocrite commits, including identifying innovative vulnerabilities and potential vulnerability-introducing minor patches. We also identify multiple factors that can increase the probability of hypocrite commits and render the patch-review process less effective. As proof of concept, we take the Linux kernel as target OSS and subtly demonstrate that it is practical for a malicious contributor to introduce insecure/buggy fixes. Furthermore, we systematically measure and characterize the capabilities and opportunities of a malicious contributor. At last, to improve the security of OSS, we propose mitigation against hypocrite commits, such as updating the list of commit for OSS and developing tools for patch testing and verification.

I. INTRODUCTION

Open source software (OSS) shares its source code publicly, and allows users to see, modify, and even distribute under an open-sourcing license. Since the founding of the Open Source Initiative in 1998, OSS has thrived and become quite popular. For example, as of August 2020, GitHub was reported to have over 40 million users and more than 27.6 million public repositories [1] (increased by 10 million from June 2018 [1]). It was also reported that everyone sees OSS [30] while 79% of computers use OSS [36]. OSS is praised for its unique advantages. The availability and low costs of OSS enable its quick and wide adoption.

Its openness also encourages contributions; OSS typically has thousands of independent programmers testing and fixing bugs of the software. Such an open and collaborative development not only allows higher flexibility, transparency, and quicker evolution, but is also believed to provide higher reliability and security [21].

A prominent example of OSS is the Linux kernel, which is one of the largest open-source projects—more than 20 million lines of code used by billions of devices. The Linux kernel involves more than 22K contributors. Any person or company can contribute to its development, e.g., submitting a patch through git commits. To make a change of the Linux kernel, one can email the patch file (containing git diff information) to the Linux community. Each module is assigned with a few maintainers (the list can be obtained through the script `get_maintainer.pl`). The maintainers then manually or employ tools to check the patch and apply it if it is deemed valid. Other popular OSS, such as FreeBSD, Firefox, and OpenSSL, also adopt a similar patching process.

Because of the wide adoption, OSS like the Linux kernel and OpenSSL, has become attractive targets for high-profile attacks [9, 15]. While adversaries are incentivized, it is not always easy to find an exploitable vulnerability. Popular OSS is often extensively tested by developers and users in both static and dynamic ways [41]. Even a bug was found, it may not maximize the exploitability and impact as the adversaries wish. Thus, finding ideal exploitable vulnerabilities requires not only advanced analysis and significant efforts, but also a bit of luck.

In this paper, we instead investigate the insecurity of OSS from a critical perspective—the feasibility of a malicious contributor stealthily introducing vulnerabilities such as use-after-free (UAF) in OSS through hypocrite commits (seemingly beneficial minor commits that actually introduce other critical issues). Such introduced vulnerabilities can be critical: they can exist in the OSS for a long period and be exploited by the malicious contributor to impact a massive number of devices and users. Specifically, we conduct a set of studies to systematically understand and characterize hypocrite commits, followed by our mitigation for mitigating such malicious contributors. We first identify three fundamental reasons that allow the hypocrite commits.

• OSS openness. By its nature, OSS typically allows anyone

S & P Home Program Call For... Attend Workshops

MAY 24-27, 2021

42nd IEEE Symposium on Security and Privacy

Sponsored by the IEEE Computer Society Technical Committee on Security and Privacy in cooperation with the International Association for Cryptologic Research

Registration and Access

Accepted Papers

A Decentralized and Encrypted National Gun Registry

Seny Kamara (Brown University), Tarik Moataz (Aroki Systems), Andrew Park (Brown University), Lucy Qin (Brown University)

A First Look at Zoombombing

Chen Ling (Boston University), Utkucan Balci (Binghamton University), Jeremy Blackburn (Binghamton University), Gianluca Stringhini (Boston University)

A Novel Dynamic Analysis Infrastructure to Instrument Untrusted Execution Flow Across User-Kernel Spaces

Jiaqi Hong (Singapore Management University), Xuhua Ding (Singapore Management University)

A Secure and Formally Verified Linux KVM Hypervisor

Shih-Wei Lu (Columbia University), Xupeng Lu (Columbia University), John Hui (Columbia University), Jason Nieh (Columbia University), Ronghui Gu (Columbia University)

<https://github.com/QiushiWu/CISS2021>

2020-12-01: Sarah Jamie Lewis & others send a letter to IEEE S&P

To: sp21-pcchairs@ieee-security.org, a.oprea@northeastern.edu, thorsten.holz@rub.de
CC: rm@ins.jku.at, santiagotorres@purdue.edu, sarah@openprivacy.ca
Subject: Ethics concerns about paper accepted for publication

Dear IEEE S&P PC Chairs,

We are a group of security professionals and academics who are reaching out with concerns about the ethics on a paper that was recently accepted to appear in IEEE S&P 2021. The paper in question is "Open Source Insecurity: Stealthily Introducing Vulnerabilities via Hypocrite Commits", and we were made aware of its acceptance through Twitter by one of the authors (likely the PI of the research group). To our dismay, it appears that the research did perform experiments with human subjects (open source developers), yet there is no indication that the experiment itself went through any type of IRB approval process². To further complicate things, it appears that the experiment has dire consequences: introducing exploitable code in perhaps the most fundamental part of computing infrastructure today – the Linux kernel.

We are reaching out because we believe this is a delicate circumstance that may taint the relationship between academics and open source professionals everywhere. If researchers are not trusted within open source circles, research relationships, experiments and dissemination of intellectual work will be affected. This is not to mention that open source has historically been a common medium for academic research to have broader impact.

<https://hackmd.io/s/BJGs6Tfiw>



Sarah Jamie Lewis
@SarahJamieLewis
Executive Director @OpenPriv.

Enforcing Consent & Resisting Surveillance with Cryptography. Vegan Lesbian, Queer Anarchist.

Donator: openprivacy.ca

© Unceded territories of the xwma8kwayam (Musqueam), Skwxwú7mesh (Squamish), Sto:lo and Salilwata7/Selilwitulh (Tsleil-Waututh) People
sarahjamielewis.com 📅 2007年7月からTwitterを利用しています

519 フォロワー中 3.2万 フォロワー
フォローしている人にフォロワーはいません

ツイート ツイートと返信 メディア いいね

↓ 固定されたツイート

Sarah Jamie Lewis @SarahJamieLewis · 2018年3月30日
Hi I'm Sarah, Executive Director of Open Privacy (@OpenPriv) - A Canadian non-profit society dedicated to researching & building privacy enhancing tools that empower people & marginalized communities.

Donate to support our work:

2020-12-15: UMN が本研究の内容についての説明をFAQとして公表

Clarifications on the “hypocrite commit” work (FAQ)

Qiushi Wu and Kangjie Lu, *University of Minnesota*

December 15, 2020

We recently finished a work that studies the patching process of OSS. Its goal is to improve the security of the patching process. The corresponding paper has been accepted by IEEE S&P 2021. I shared the abstract of the paper on Twitter, which then resulted in heated discussion and pushback. I apologize for the misleading abstract which did not show the details and caused many confusions and misunderstandings. Therefore, we would like to make a few clarifications.

We would like to first mention that we are a young research group with improving the kernel security as the first priority. In the past several years, we devote most of our time to improving the Linux kernel, and we have found and fixed more than one thousand kernel bugs; the extensive bug finding and fixing experience also allowed us to observe issues with the patching process and motivated us to improve it. Thus, we consider ourselves security researchers as well as OSS contributors. We respect OSS volunteers and honor their efforts. We have never intended to hurt any OSS or OSS users. We did not introduce or intend to introduce any bug or vulnerability in OSS. The following are the clarifications to the common concerns we received.

* The purpose and research value of the work

The project aims to improve the security of the patching process in OSS. As part of the project, we study potential issues with the patching process of OSS, including causes of the issues and suggestions for addressing them. This study indeed reveals some issues, but its goal is to call for efforts to improve the patching process—to motivate more work that develops techniques to test and verify patches, and finally to make OSS safer.

In this work, we collect 138 previous bug-introducing patches (not introduced by us). Based on these patches, we summarize their patterns, study specific reasons why bug-introducing patches are hard to catch (with both a qualitative and a quantitative analysis), and more importantly, provide suggestions to addressing the problem. In this work, we introduce the concept of “innate vulnerability” where a vulnerability condition of it is missing, but it can be turned into a real one when the condition is implicitly introduced by a patch for another bug. We also develop tools that help us find code places that may suffer from bug-introducing patches, and suggest what may make these bug-introducing patches hard to catch.

* Did the authors introduce or intend to introduce a bug or vulnerability?

No. As a part of the work, we had an experiment to demonstrate the practicality of bug-introducing patches. This is actually the major source of the raised concerns. In fact, this experiment was done safely. We did not introduce or intend to introduce any bug or vulnerability in the Linux kernel. All the bug-introducing patches stayed only in the email exchanges, without being adopted or merged into any Linux branch, which was explicitly confirmed by maintainers. Therefore, the bug-introducing patches in the email did not even become a Git commit in any Linux branch. None of the Linux users would be affected. The following shows the specific procedure of the experiment.

OSS パッチ投稿メカニズムに内在するリスクの顕在化

- 過去 138 件のパッチがバグを作り込んだ事実を把握
- Linux kernel にバグを入れる事が目的ではない
- メンテナー自体が研究対象ではない (UMN/IRB 裁定)
- コミュニティには論文投稿前に 一般論としての問題指摘を実施した (が、Hypocrite Commits 実験を行うことをメンテナーに事前告知しなかった [筆者注])
- 本研究がメンテナーの貴重な時間を消費させる事は認識
- コミュニティとの関係を崩すことにはならないと期待

<https://www-users.cs.umn.edu/~kjl/papers/clarifications-hc.pdf>

UMN Institutional Review Board (研究倫理審査委員会) の Go 判断

* Is this human research?

This is not considered human research. This project studies some issues with the patching process instead of individual behaviors, and we did not collect any personal information. We send the emails to the Linux community and seek community feedback. The study does not blame any maintainers but reveals issues in the process. The IRB of UMN reviewed the study and determined that this is not human research (a formal IRB exempt letter was obtained).

Throughout the study, we honestly did not think this is human research, so we did not apply for an IRB approval in the beginning. We apologize for the raised concerns. This is an important lesson we learned---Do not trust ourselves on determining human research; always refer to IRB whenever a study might be involving any human subjects in any form. We would like to thank the people who suggested us to talk to IRB after seeing the paper abstract.

<https://www-users.cs.umn.edu/~kjl/papers/clarifications-hc.pdf>



Calm Down

以降 UMN とコミュニティの会話が途切れ 事態は沈静化したかにみえたが....

2021-04-06: Aditya Pakki @ UMN のパッチから 大炎上に発展

```
From Aditya Pakki <>
Subject [PATCH] SUNRPC: Add a check for gss_release_msg
Date Tue, 6 Apr 2021 19:16:56 -0500
```

In gss_pipe_destroy_msg(), in case of error in msg, gss_release_msg deletes gss_msg. The patch adds a check to avoid a potential double free.

Signed-off-by: Aditya Pakki <pakki001@umn.edu>

```
---
net/sunrpc/auth_gss/auth_gss.c | 3 ++
1 file changed, 2 insertions(+), 1 deletion(-)
```

```
diff --git a/net/sunrpc/auth_gss/auth_gss.c b/net/sunrpc/auth_gss/auth_gss.c
index 5f42aa5fc612..eb52ebb3923 100644
--- a/net/sunrpc/auth_gss/auth_gss.c
+++ b/net/sunrpc/auth_gss/auth_gss.c
@@ -848,7 +848,8 @@ gss_pipe_destroy_msg(struct rpc_pipe_msg *msg)
    warn_gssd();
    gss_release_msg(gss_msg);
}
- gss_release_msg(gss_msg);
+ if (gss_msg)
+   gss_release_msg(gss_msg);
}

static void gss_pipe_dentry_destroy(struct dentry *dir,
--
2.25.1
```

<https://lkml.org/lkml/2021/4/6/1295>

```
Date Tue, 20 Apr 2021 09:15:23 +0200
```

```
From Greg KH <>
```

```
Subject Re: [PATCH] SUNRPC: Add a check for gss_release_m
```

On Tue, Apr 06, 2021 at 07:16:56PM -0500, Aditya Pakki wrote:

If you look at the code, this is impossible to have happen.

Please stop submitting known-invalid patches. Your professor is playing around with the review process in order to achieve a paper in some strange and bizarre way.

This is not ok, it is wasting our time, and we will have to report this, AGAIN, to your university...

greg k-h

2021-04-21: Greg が MNU からのパッチ拒否、過去パッチも全削除

You, and your group, have publicly admitted to sending known-buggy patches to see how the kernel community would react to them, and published a paper based on that work.

Now you submit a new series of obviously-incorrect patches again, so what am I supposed to think of such a thing?

They obviously were `_NOT_` created by a static analysis tool that is of any intelligence, as they all are the result of totally different patterns, and all of which are obviously not even fixing anything at all. So what am I supposed to think here, other than that you and your group are continuing to experiment on the kernel community developers by sending such nonsense patches?

When submitting patches created by a tool, everyone who does so submits them with wording like "found by tool XXX, we are not sure if this is correct or not, please advise." which is NOT what you did here at all. You were not asking for help, you were claiming that these were legitimate fixes, which you KNEW to be incorrect.

<https://lkml.org/lkml/2021/4/21/143>

2021-04-21: Greg が MNU からのパッチ拒否、過去パッチも全削除

A few minutes with anyone with the semblance of knowledge of C can see that your submissions do NOT do anything at all, so to think that a tool created them, and then that you thought they were a valid "fix" is totally negligent on your part, not ours. You are the one at fault, it is not our job to be the test subjects of a tool you create.

Our community welcomes developers who wish to help and enhance Linux. That is NOT what you are attempting to do here, so please do not try to frame it that way.

Our community does not appreciate being experimented on, and being "tested" by submitting known patches that are either do nothing on purpose, or introduce bugs on purpose. If you wish to do work like this, I suggest you find a different community to run your experiments on, you are not welcome here.

Because of this, I will now have to ban all future contributions from your University and rip out your previous contributions, as they were obviously submitted in bad-faith with the intent to cause problems.

plonk

greg k-h

Plonk (Usenet)

From Wikipedia, the free encyclopedia

For other uses, see [Plonk](#).

Plonk is a [Usenet](#) jargon term for adding a particular poster to one's [kill file](#) so that poster's future postings are completely ignored. It was first used in 1989, and by 1994 was a commonly used term on Usenet.^[1] To publicly repudiate a poster, it is added to one's reply or is simply used as the entire, one-word reply. It may also be used as a verb. The word is an example of [onomatopoeia](#), intended to represent the metaphorical sound of the plonked user hitting the bottom of the kill file.^[2]

[Folk etymology](#) sometimes gives the term's origin as an [acronym](#) of various phrases, although these are likely to be [backronyms](#). These backronyms include: Please Log Off, Net Kook; Put Lamer On Killfile,^[3] and Please Leave Our Newsgroup: Killfile!^[*citation needed*] The term's usage later expanded to include blocking messages from annoying senders by using [e-mail filters](#) that delete incoming messages based on criteria set by the email recipient. Plonk has similarly been used on [BBSes](#), [online forums](#), [blogs](#), [IRC \(Internet Relay Chat\)](#), and [wikis](#) (which usually do not have filters). It is occasionally used in reference to blocking a user on [instant messaging \(IM\)](#) or a [social media](#) site.

[https://en.wikipedia.org/wiki/Plonk_\(Usenet\)](https://en.wikipedia.org/wiki/Plonk_(Usenet))

公知化=Linux コミュニティで起こった大事件 として拡散 (4/22)

Linuxカーネルに意図的にバグを混入したとして大学にコミュニティ出禁措置



オープンソースソフトウェアの脆弱(ぜいじゃく)性に関する論文の執筆のため、Linuxカーネルに既知のバグを含むパッチを送信したことを理由に、ミネソタ大学に対して「Linuxカーネル開発への貢献の禁止」、つまり出禁措置が行われました。

Linux bans University of Minnesota for sending buggy patches in the name of research [Update] - Neowin

<https://www.neowin.net/news/linux-bans-university-of-minnesota-for-sending-buggy-patches-in-the-name-of-research/>

<https://gigazine.net/news/20210422-linux-ban-university/>

実際にはコミュニティ幹部は極めて冷静かつ合理的に迅速に対処



ミネソタ大研究者、研究のためとしてLinuxカーネルに意図的に脆弱性コードをコミット

ストーリー by nagazou 2021年04月26日 12時00分 疑心暗鬼発生 部門より

4月21日、Linuxカーネルの開発コミュニティで、ミネソタ大学の研究者らがLinuxカーネルのソースコードに既知のセキュリティ上の欠陥のあるコードをコミットしていたとして、Linuxカーネルへの貢献を禁止する処置が行われたことが話題になっている ([lore.kernel.orgのLKMLアーカイブ](https://lore.kernel.org/LKML/)、[The Verge](https://www.theverge.com/2021/4/21/22361114/linux-kernel-security-vulnerability)、[Phoronix](https://www.phoronix.com/scan.php?page=news_item&id=Linux-Kernel-Exploit)、[GIGAZINE](https://www.gigazine.net/news/20210421-linux-kernel-vulnerability/)、[ITmedia](https://www.itmedia.co.jp/news/2021/04/21/22361114.html))。

この問題に関与した研究者は論文を発表し、カーネル開発コミュニティが、悪意あるコードを変更を審査する能力があるかどうかを試すために、意図的に実行したものだとしている ([GitHub 論文\[PDF\]](#))。コミュニティは同大学からの新しいコードを受け入れないことに加え、過去に提出されたすべてのコードを削除、再審査しているという。開発者コミュニティ側の[Greg Kroah-Hartman](#)は、我々の時間を無駄にする行為だとして批判している。

なお、Tom's Hardwareの記事によれば、[Linus Torvalds氏](#)の反応は「私は本当に何を言うべきかわからない。技術的には大したことではないと思うが、コミュニティの人々は腹を立てており、明らかに信頼を侵害したと思う」と予想よりも穏やかなものであったらしい ([Tom's Hardware](#))。

あるAnonymous Coward曰く、

「ポストモダン思想におけるソーカル事件みたいなことを再現したかったのかな、とも思うが、影響範囲がひどいからやってはダメ。ぜったい。」

[情報元へのリンク](#)

130 コメント linux



<https://linux.srad.jp/story/21/04/25/1954223/>

総括、何が問題だったのか

本事案の 公知化以降の急展開 ① (4/21 - 4/23)

Greg の警告直後から 事態は急展開 していった

- 4/21: LF TAB (Technical Advisory Board) が Hypocrite Commits 事案の調査を開始
- 4/21: 偽装送信者特定 (James Bond,..)
- 4/21: Hypocrite Commits パッチの特定
- 4/21: UMN Department Head が謝罪文 (→)
- 4/22: LF TAB が最初のレビューコメントを発表
- 4/23: LF が UMN に 改善命令レター を発行

[Home](#) > [Statement from CS&E on Linux Kernel research - April 21, 2021](#)

Statement from CS&E on Linux Kernel research - April 21, 2021

Leadership in the University of Minnesota Department of Computer Science & Engineering learned today about the details of research being conducted by one of its faculty members and graduate students into the security of the Linux Kernel. The research method used raised serious concerns in the Linux Kernel community and, as of today, this has resulted in the University being banned from contributing to the Linux Kernel.

We take this situation extremely seriously. We have immediately suspended this line of research and will investigate the research method and the process by which this research method was approved to determine appropriate remedial action, and safeguard against future issues, if needed. We will share our findings back to the community as soon as practical.

Sincerely,

Mats Heimdahl, Department Head
Loren Terveen, Associate Department Head

Greg の警告に対してコミュニティと UMN は間髪を入れずアクションを開始する

本事案の 公知化以降の急展開 ② (4/14 - 5/5)

UMN は 本事案の調査に全面協力、公式謝罪へ

- 4/24: Kanjie Lu 助教授が **LKML に謝罪文** 投稿
- 4/25: UMN が一連の Commits メールを収集
- 4/26: UMN が IEEE に **投稿論文の取り下げ要請** (→)
- 4/27: UMN が **詳細調査報告** を公表
- 4/27: UMN が **LF の改善命令** に返信
- 5/5: LF TAB が **詳細レポート公開**
 - UMN 投稿パッチの全件レビュー結果を含む

April 26, 2021

Qiushi Wu, Kangjie Lu
University of Minnesota

Dear Professor Oprea and Professor Holz:

We wish to withdraw our paper "On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits" from publication in the 42nd IEEE Symposium on Security and Privacy.

We are making this decision because we realized that our study design was inappropriate: specifically, it involved conducting research on the Linux kernel community without obtaining appropriate consent and approval.

More specifically, we are withdrawing the paper for two reasons:

First, we made a mistake by not engaging in collaboration with the Linux kernel community before conducting our study. We now understand that it was inappropriate and hurtful to the community to make it a subject of our research, and to waste its effort reviewing these patches without its knowledge or permission. Instead, we now realize that the appropriate way to do this sort of work is to engage with community leaders beforehand so that they are aware of the work, approve its goals and methods, and can support the methods and results once the work is completed and published. Therefore, we are withdrawing the paper so that we do not benefit from an improperly conducted study.

Second, given the flaws in our methods, we do not want this paper to stand as a model for how research can be done in this community. On the contrary, we hope this episode will be a learning moment for our community, and that the resulting discussion and recommendations can serve as a guide for proper research in the future. Therefore, we are withdrawing the paper to prevent our misguided research method from being seen as a model for how to conduct studies in the future.

We sincerely apologize for any harm our research group did to the Linux kernel community, to the reputation of the IEEE Symposium on Security and Privacy, our Department and University, and our community as a whole.

Sincerely,

Qiushi Wu and Kangjie Lu

オープンソースだったから「極めて迅速で完全に透明性のある分析」が可能であった

本事案の 公知化以降の急展開 ③ UMN 投稿パッチレビュー

Greg によるパッチレビュー経緯

- 4/21: 過去パッチ revert 開始
- 4/28: Hypocrite Commits revert
- 4/29: パッチ再レビュー結果投稿
- 5/3: 最終 revert パッチ投稿

- 信頼されたメンテナーが多数参加
- LF/TAB で最終判定
- 5/3: **パッチレビュー結果投稿 (→)**

レビュー結果 = 大部分は正当な修正

- UMN 過去パッチ全件調査 (対象 435 件)
 - **正当性を確認 (349 件)**
 - 修正が必要 (39 件)
 - 後のコミットで修正済 (25 件)
 - 既に解決済みで不要 (12 件)
 - 研究グループ結成前 (9 件)
 - 作者から revert 要求 (1 件)

- Hypocrite Commits (対象 5 件)
 - 脆弱性付のパッチ (4 件)
 - **間違えて正しいパッチ (1 件)**

Lessons and Learned

Hypocrite Commits 結果

- メンテナーにより Reject されて
一件もマージされなかった
- 脆弱性の検出云々以前の問題として
パッチとして成立していなかった から
- UMN 指摘のリスクは否定していない
 - 「予防的パッチの適用」は実績多数
 - 「ツールの活用」も既に実施中
 - 脆弱性混入リスクは排除できない
 - 脆弱性対策については有効に対処中

Hypocrite Commits の教訓

- 実験台にされた事実に強い抵抗感
- アカデミアとの信頼関係に大きな傷
- 過去には良好な協調関係があった
- Linux 誕生はヘルシンキ大学に遡る
- アカデミアとの 信頼関係再構築を期待
- 今後のアカデミアの研究活動の指針
となる事件になった

セキュリティ問題深刻化、OSS 適用範囲拡大に対してどう連携するかは重要な課題

Reference

今日説明の UMN 関連の内容は以下 LKML から参照可能

- 本事案に対するコミュニティ総括 [LKML]
- <https://lkml.org/lkml/2021/5/5/1244>
- 投稿論文
- <https://github.com/QiushiWu/QiushiWu.github.io/blob/main/papers/OpenSourceInsecurity.pdf>

<https://github.com/gregkh/presentation-linux-maintainer> ①

Linux's response

- Kernel fixes available on announcement date
- Intel notified some kernel developers in advance
- Worked together across OS vendors to solve
- Much better than Spectre/Meltdown
- Process still needs to improve, Debian notified 48 hours before release.
- More fixes came after announcement
- Update your kernel and BIOS!



<https://github.com/gregkh/presentation-linux-maintainer> ②

Linux security fixes

- Happen at least once a week
- Look like any other bugfix
- Rarely called out as security fix
- Many bugfixes not known to be security related until years later
- No differentiation between bug types
 - A bug is a bug is a bug
- Very few CVEs ever get assigned for kernel security issues



<https://github.com/gregkh/presentation-linux-maintainer> ③

Linux security fixes != CVEs

- Small fraction of kernel security fixes get CVEs
- If you only cherry-pick CVEs, you have an insecure system
- Some CVEs have follow-on fixes not documented anywhere
- [How the Linux Kernel Security team works](#)



<https://github.com/gregkh/presentation-linux-maintainer> ④

Linux security fixes != CVEs

- Small fraction of kernel security fixes get CVEs
- 2006-2018 had 1005 CVEs assigned to the kernel
 - 41% (414) had a negative “fix date”
 - 12 never fixed
 - Average fix date, -100 days
 - Longest fix dates, -3897 and 2348 days
 - 88 fixed within 1 week
 - Standard deviation 405



<https://github.com/gregkh/presentation-linux-maintainer> ⑤

Linux Longterm Kernels Fix Problems

- Bugs are fixed before you realize it is a issue.
- Google security team requests for Pixel phones in 2018:
 - 92% (201/218) problems were already fixed in LTS kernel
 - No need for cherry-picking or backporting
 - Remaining issues were due to out-of-tree code

